Self-Adaptive Discovery Mechanisms for Improved Performance in Fault-Tolerant Networks

Kevin Mills, Chris Dabrowski, and Jesse Elder

DARPA FTN PI Meeting

January 18, 2002

Survivable Software for Harsh Environments

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **18 JAN 2002**	2. REPORT TYPE	3. DATES COVERED **00-00-2002 to 00-00-2002**
4. TITLE AND SUBTITLE **Self-Adaptive Discovery Mechanisms for Improved Performance in Fault-Tolerant Networks**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **National Institute of Standards and Technology,Gaithersburg,MD,20899**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **presented at DARPA Fault Tolerant Network (FTN) PI Meeting, 19 Jan 2002**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**31**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Presentation Outline

- One-Page Review of Project Objective and Plan
- Brief Refresher on Service Discovery Protocols
- Outline of Technical Approach to Understand Fault-Tolerant Behavior of Service Discovery Protocols
- Initial Results Comparing Behavior of Jini and Universal Plug-and-Play when Propagating Information during Interface Failures
- Plan for Next Six Months
- Conclusions

More Details Available on Supplementary Slides

Project Objective

Research, design, evaluate, and implement self-adaptive mechanisms and algorithms to improve the performance of service discovery protocols for use in fault-tolerant networks.

Project Plan – Three Phases

- Phase I – characterize performance of selected service discovery protocols (Universal Plug-and-Play – UPnP – and Jini) as specified and implemented
  - develop simulation models for each protocol
  - establish performance benchmarks based on default or recommended parameter values and on required or most likely implementation of behaviors
- Phase II – design, simulate, and evaluate self-adaptive algorithms to improve performance of discovery protocols regarding selected mechanisms
  - devise algorithms to adjust control parameters and behavior in each protocol
  - simulate performance of each algorithm against benchmark performance
  - select most promising algorithms for further development
- Phase III – implement and validate the most promising algorithms in publicly available reference software

Dynamic Discovery Protocols in Essence

Dynamic discovery protocols enable *network elements* (including software clients and services, as well as devices):

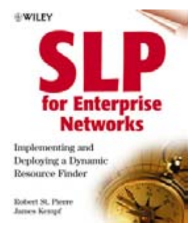
(1) to *discover* each other without prior arrangement,

(2) to *express* opportunities for collaboration,

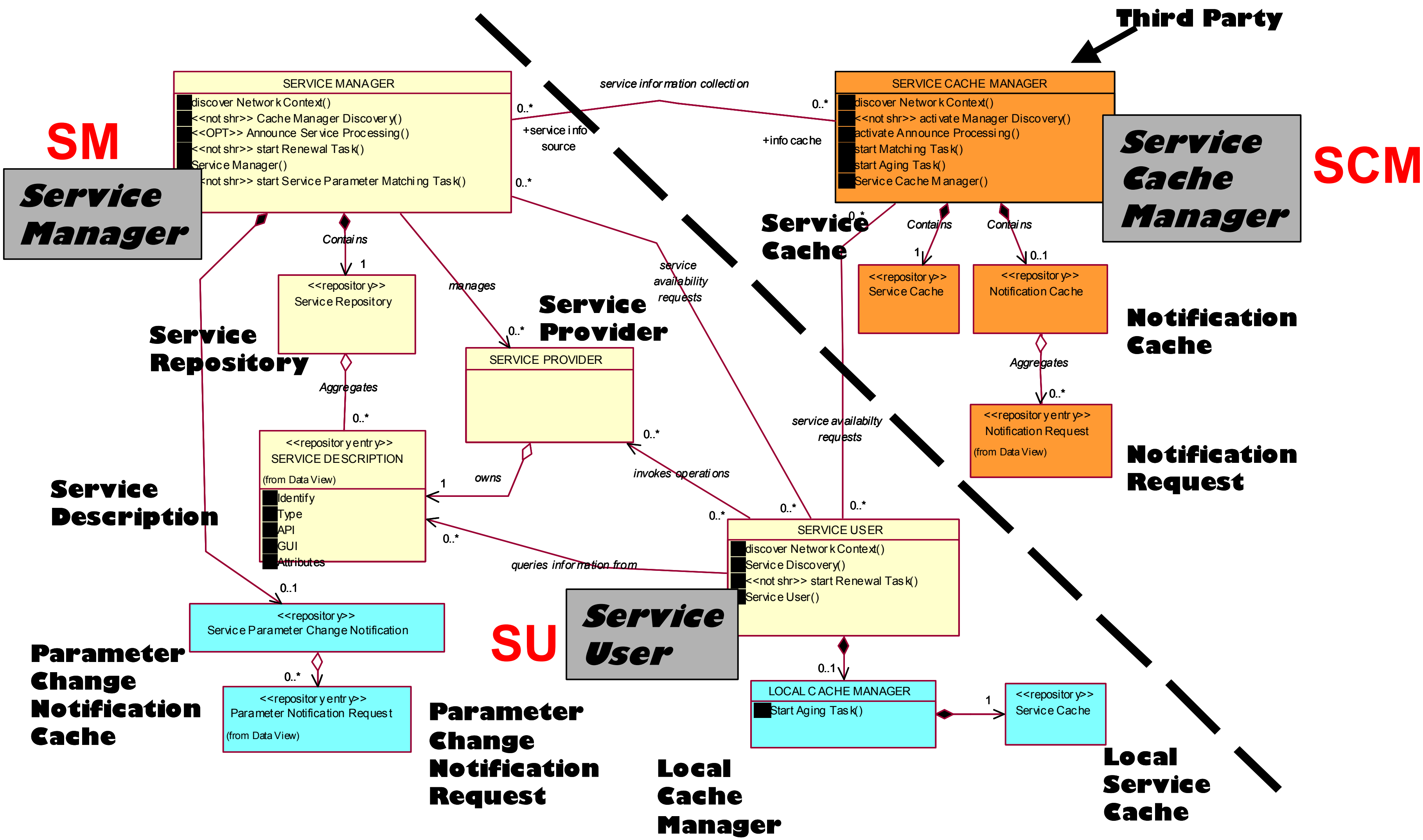
(3) to *compose* themselves into larger collections that cooperate to meet an application need, and

(4) to *detect and adapt to changes* in network topology.

Selected First-Generation Dynamic Discovery Protocols

JINI — 3-Party Design	Universal Plug and Play — 2-Party Design	 Adaptive 2/3-Party Design
The Salutation Consortium — Vertically Integrated 3-Party Design	HAVi — Network-Dependent 3-Party Design	Bluetooth™ — Network-Dependent 2-Party Design

Two Party vs. Three Party Architectures

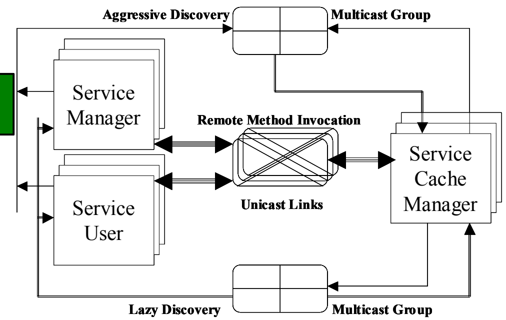


Technical Approach to Phase I – Use Rapide to Model and Understand Fault Behavior of Jini and UPnP

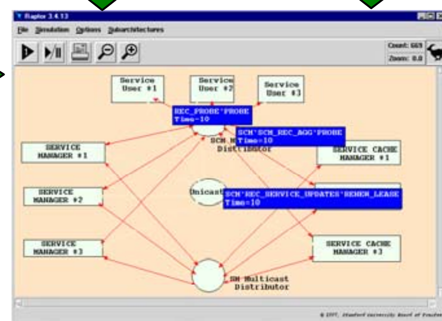
Time	Command	Parameters
5	NodeFail	SM4
5	LinkFail	SCM1 SM4
10	GroupJoin	SM4 GROUP1
10	FindService	SU8 5 1 2 S XYZ ALL
50	AddService	SM4 SCM3 T ATT API GUI 20 30

Scenario

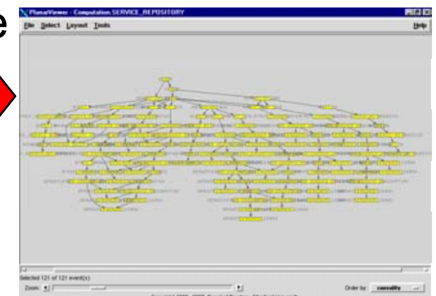
Topology



Specification Model



Execute with Rapide



```
-- ****************************************************
-- ** 3.3  DIRECTED DISCOVERY CLIENT INTERFACE      **
-- ****************************************************
-- This is used by all JINI entities in directed
-- discovery mode.  It is part of the SCM_Discovery
-- Module. Sends Unicast messages to  SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need reviw.
TYPE Directed_Discovery_Client
  (SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
   InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR    : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES      : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update   : DD_SCM_Update;
SERVICE SCM_Update      : SCM_Update;
SERVICE DB_Update       : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES;  -- events for failure and recovery.
ACTION
 IN Send_Requests(),
    BeginDirectedDiscovery();
BEHAVIOR
   action animation_Iam (name: string);
   MySourceID        : VAR IP_Address;
   PV                : VAR ProtocolVersion;
```

Consistency Conditions

For All (SM, SD, SCM):
(SM, SD) IsElementOf SCM registered-services (CC1)
implies SCM IsElementOf SM discovered-SCMs

For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs & (CC2)
(SD) IsElementOf SM managed-services
implies (SM, SD) IsElementOf SCM registered-services

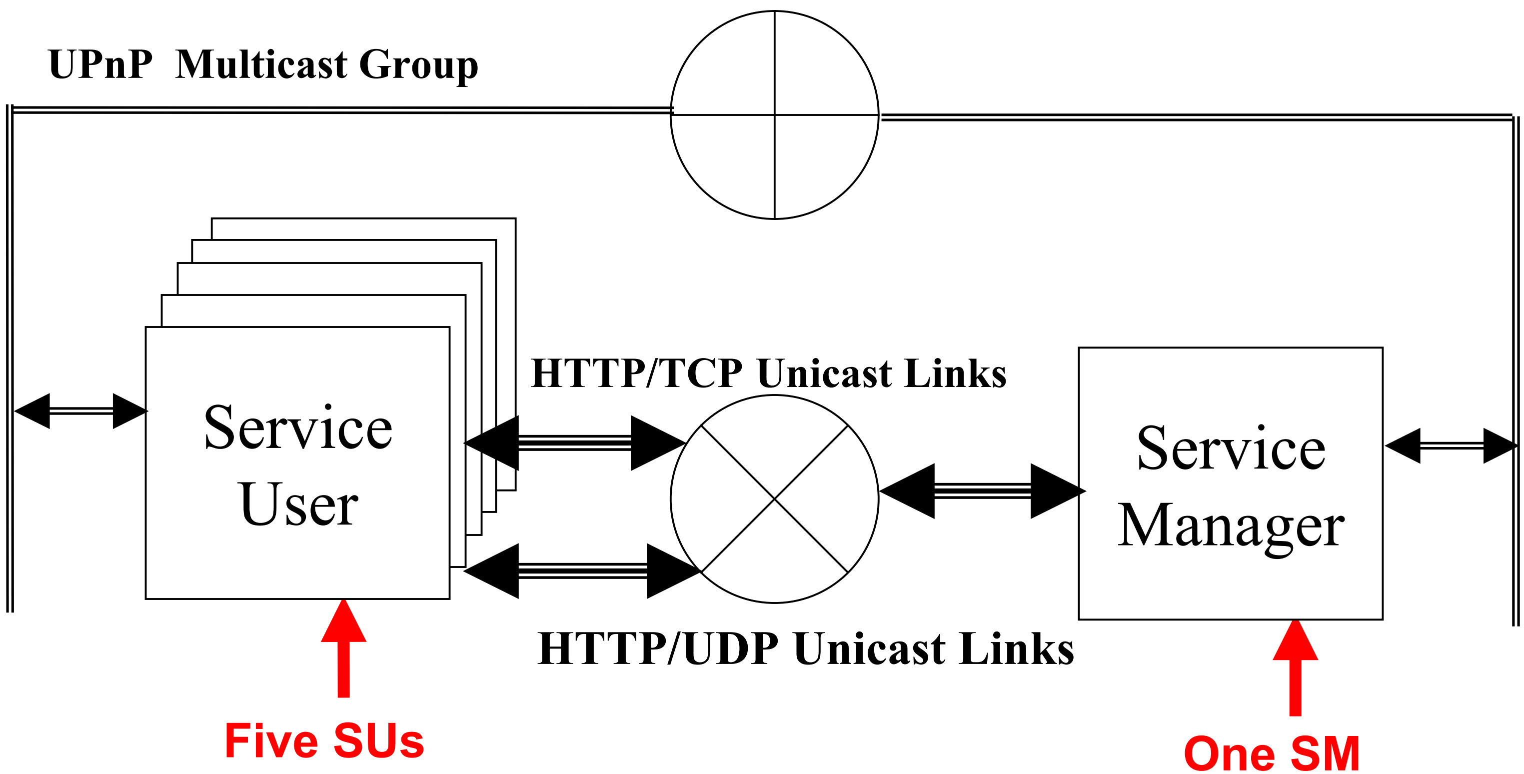
For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs & (CC3)
(SM, SD) IsElementOf SCM registered-services &
NOT (SCM IsElementOf SM persistent-list)
implies Intersection (SM GroupsToJoin, SCM GroupsMemberOf)

For All (SM, SD, SCM, SU, NR):
(SU, NR) IsElementOf SCM requested-notifications & (CC4)
(SM, SD) IsElementOf SCM registered-services &
Matches((SM, SD), (SU,NR))
implies (SM, SD) IsElementOf SU matched-services

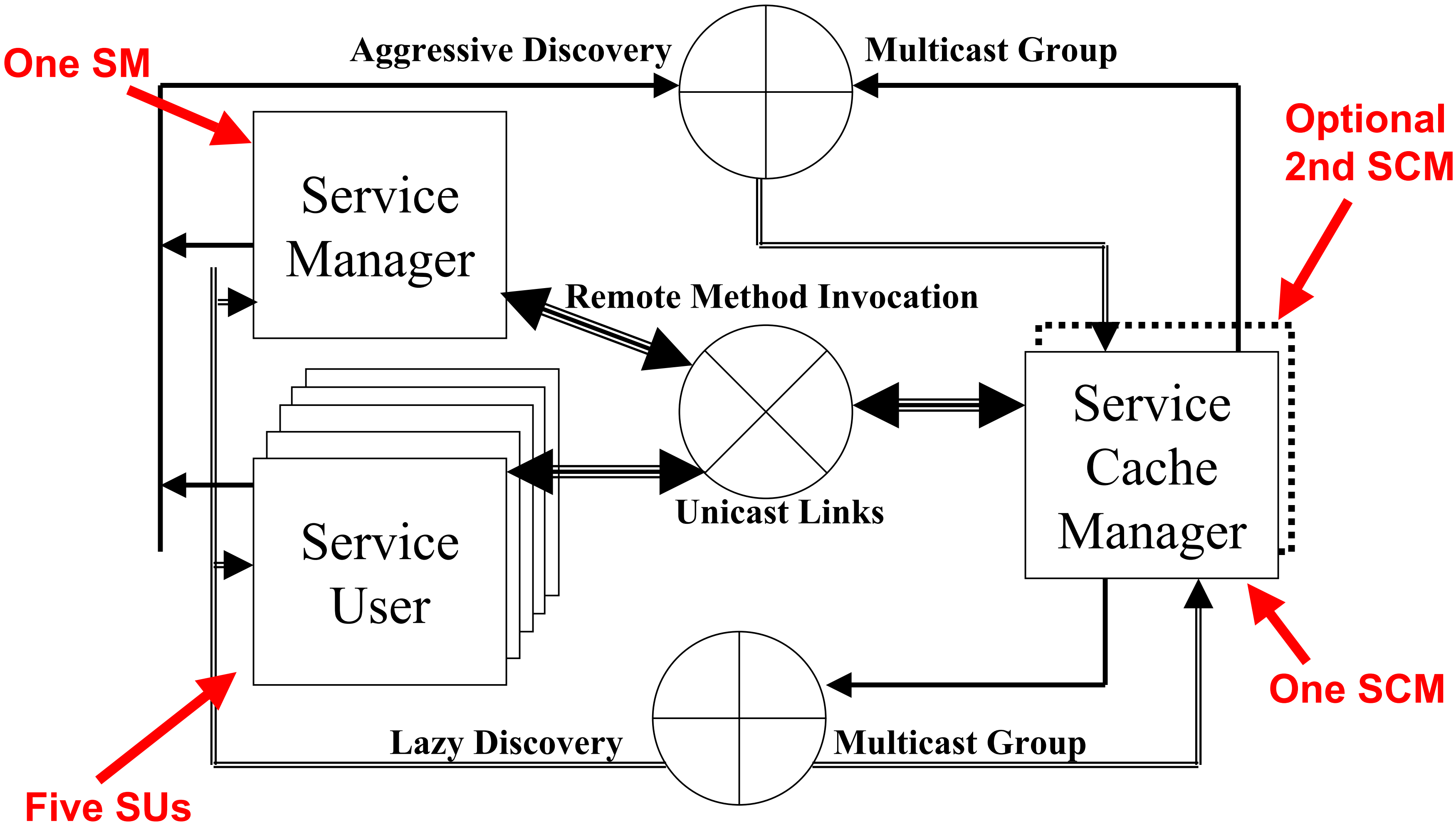
Analyze POSETs

Assess Correctness, Performance, & Complexity

Two-Party (UPnP) Topology for Experiment



Three-Party (Jini) Topologies for Experiment



Faults Interfere with Discovery and Information Propagation

- Interface Failure – Tx, Rx, or Both
  - Due to nearby enemy jamming or other interference
  - Due to multi-path fading during mobility
- Path Loss – Pt-Pt or Area-Area and Full-duplex or Half-duplex
  - Due to persistent congestion
  - Due to physical link cuts
  - Due to enemy jamming at routers
- Message Loss – under both UDP and TCP (>delay)
  - Due to sporadic or distant enemy jamming or other interference
  - Due to transient congestion
  - Due to multi-path fading during mobility
- Node Failure – Partial or Complete with variable persistence of information
  - Due to enemy bombardment or cyber attacks
  - Due to mobility associated with military operations

Discovery Systems Divide Recovery Responsibilities: Lower Layers, Discovery Protocol, and Application

- Selective Reliable Delivery by Lower Layers
  - TCP attempts retransmissions (basis for Jini-RMI and UPnP-HTTP)
  - UDP messages in UPnP sent as multiple copies
- Periodic Announcements by Discovery Protocol
  - Allows caching nodes to discard information when TTL expires
  - Jini includes aggressive search at node start up, while UPnP permits nodes to undertake aggressive search at any time
- Periodic Refreshing of Resources Required by Discovery Protocol
  - Allows resource owner to free resource when refresh period expires
- Remote Exceptions Issued by Protocol Over TCP Links
  - Allows application to take recovery action: Ignore? Retry? Discard knowledge of service or resource?

Understanding Information Propagation in Discovery Architectures during Interface Failure

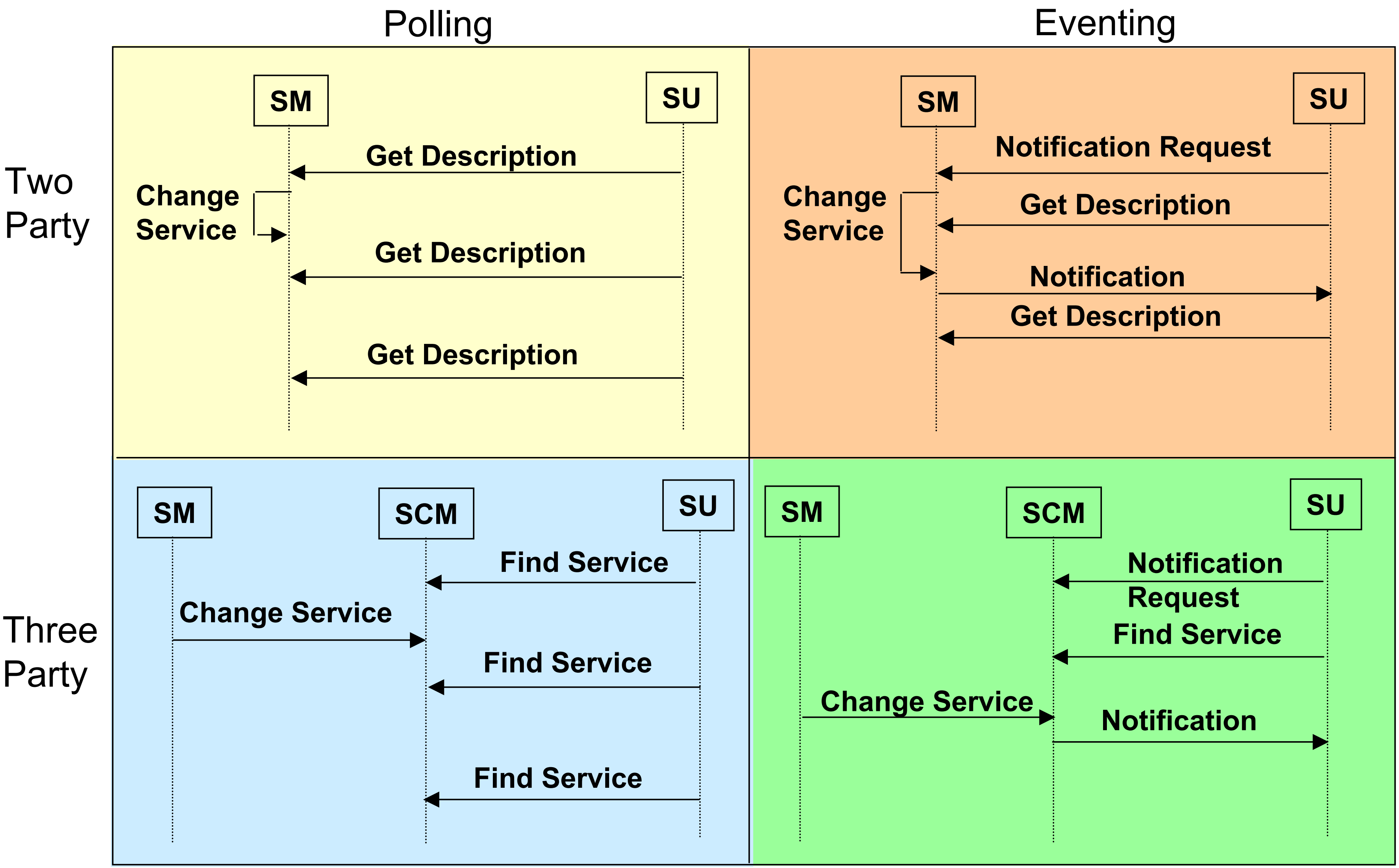
How do various service discovery architectures, topologies, and fault-recovery mechanisms perform under deadline during interface failure?

Outline of Experiment

- Deploy models of two-party (UPnP) and three-party (Jini) architectures with one SM and five SUs (for Jini include two topologies – one and two SCMs).
- Ensure initial discovery and information propagation completed.
- Introduce a change in the service description at the SM, and establish a deadline for propagating the new information to all SUs.
- Measure the number of messages exchanged and the latency required to propagate the information to all SUs, or until the deadline arrives, under two different propagation mechanisms: polling and eventing.
- Repeat this experiment while varying the percentage of interface failure time for each node up to 75% (in increments of 5%).

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

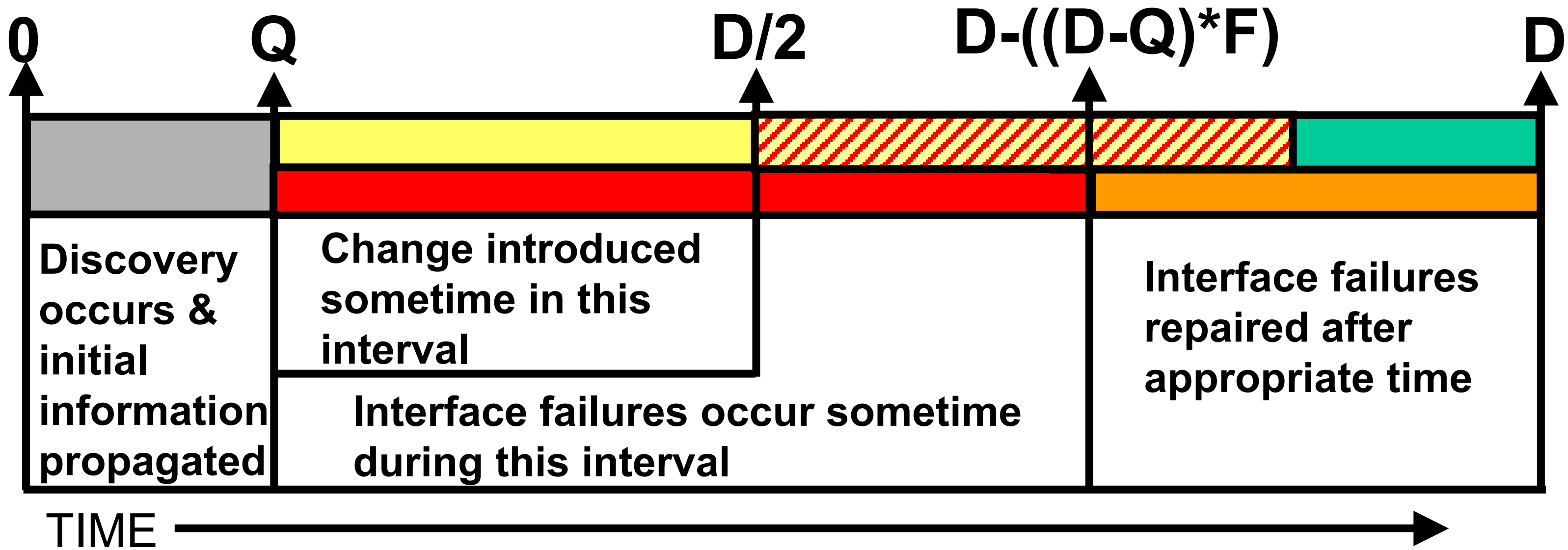
Information-Propagation Mechanisms for Experiment



NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

DARPA

Interface-Failure Model for Experiment



Random Processes

1. Choose a time to introduce the change [uniform(Q, D/2)]
2. For each node, choose a time to introduce an interface failure [uniform(Q, D-((D-Q)*F))]
3. When each interface failure occurs, choose the scope of the failure, where each of [Rx, Tx, Both] has an equal probability

Q = end of quiescent period (100 s in our experiment)
D = propagation deadline (5400 s in our experiment)
F = Interface Failure Rate (variable from 0% - 75% in 5% increments in our experiment)

Our Model Responses to Remote Exceptions ("Approximately")

- Ignore REX Received
  - When replying to a remote-method invocation
  - When attempting to cancel a lease
  - When attempting to renew a lease (But then attempt to obtain a new lease)
- Retry Operation for Some Period of Time - Then Quit If Not Successful (If Quitting, Eliminate Local Knowledge of Discovered Entity)
  - When attempting to register for notification events
  - When a UPnP SU requests service descriptions
- Retry Operation Persistently as Long as Application Executes
  - When a Jini SM attempts to register a service description with a SCM

Metrics Devised for Information-Propagation Experiments

- Update Responsiveness (*R*)

  Assuming that information is created at a particular time and must be propagated by a deadline, then the difference between the deadline and the creation time represents available time in which to propagate the information. Update Responsiveness, *R,* measures the *proportion of the available time remaining after the information is propagated*. [1 = all time remains and 0 = no time remains]

- Update Effectiveness (*U*)

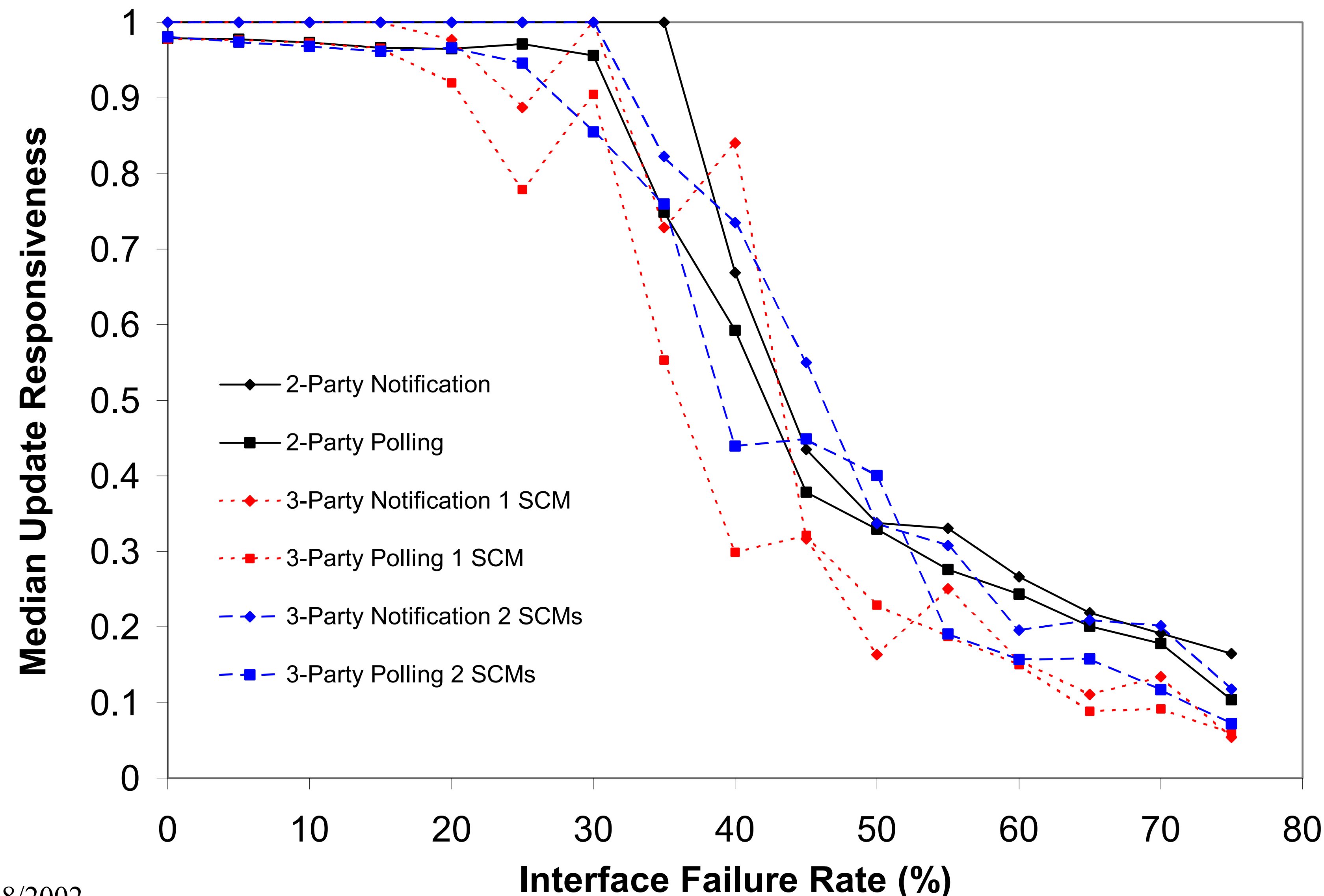
  Update Effectiveness, *U*, measures the *probability that information will propagate successfully to a SU before some deadline*, *D*. [1 = information will be propagated and 0 = information will not be propagated]

- Update Efficiency (*E*)

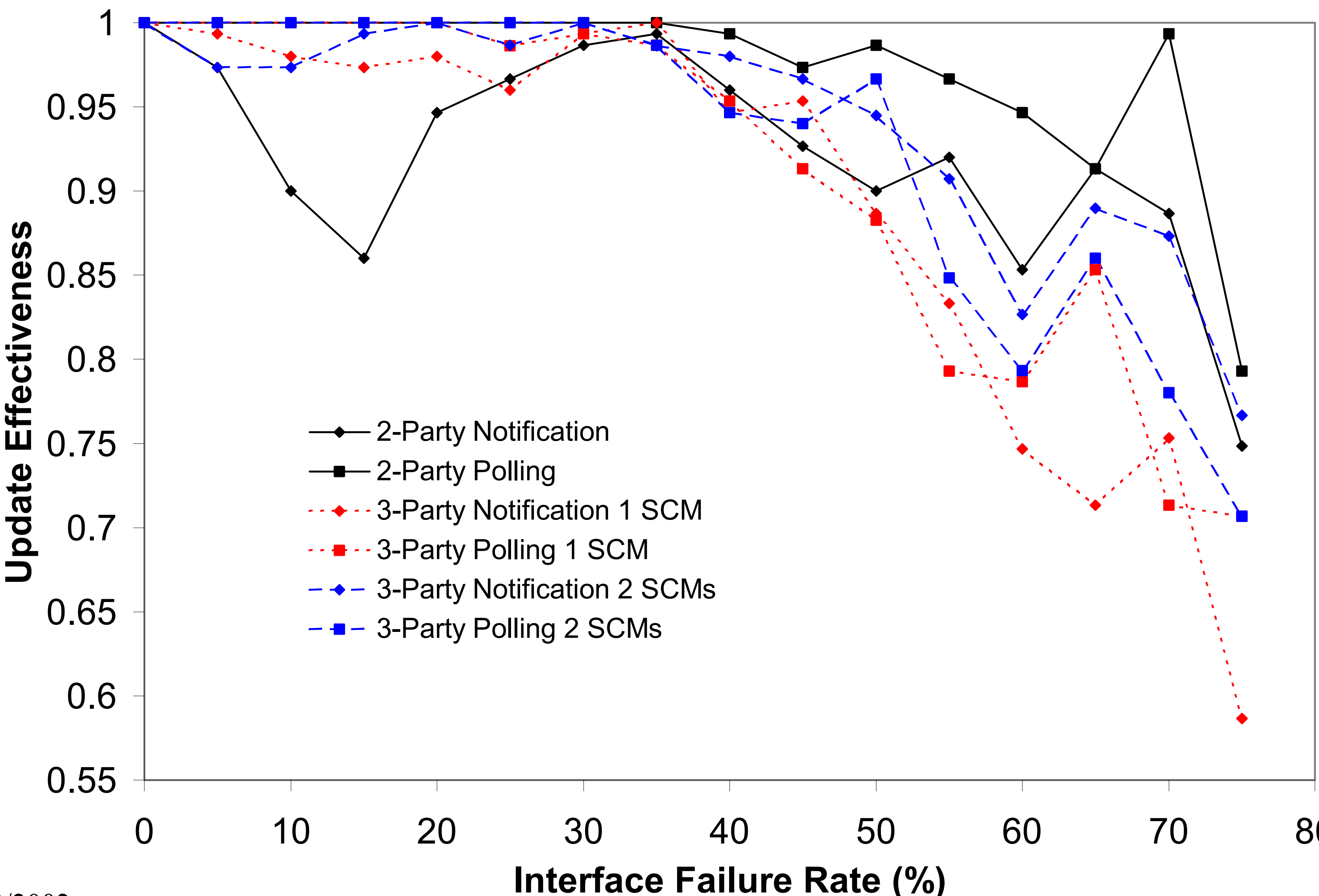
  Given a specific topology of SUs and SMs in a discovery system, examination of the available architectures (two-party and three-party) and mechanisms (polling and eventing) will reveal a minimum number of messages that need to be sent to propagate information from all SMs to all SUs in the topology. Update Efficiency, *E*, can be measured as the *ratio of the minimum number of messages needed to the actual number of messages observed*. [1 = only minimum number of messages needed and 0 = infinite number of messages needed]

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

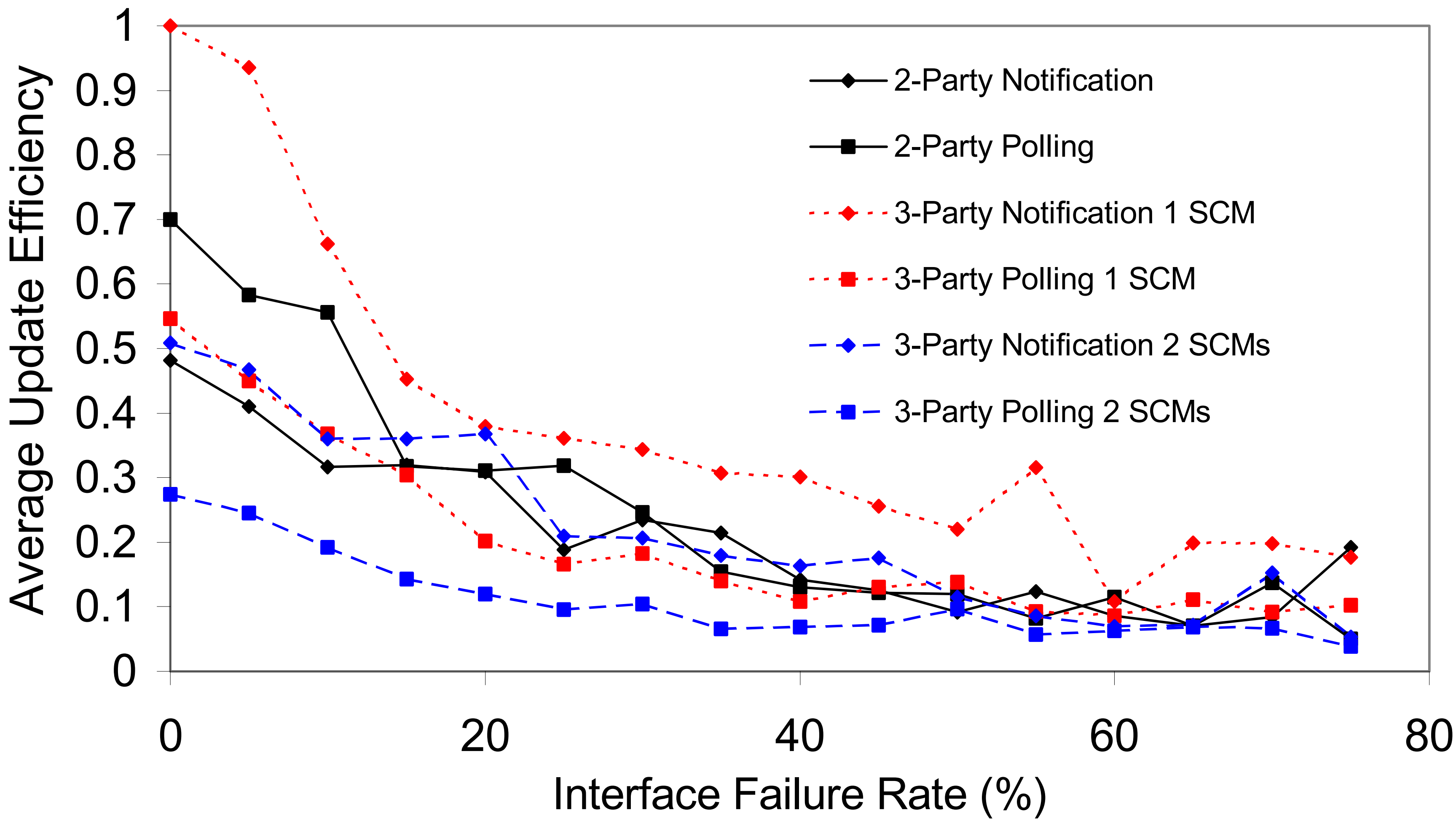
Responsiveness: UPnP (2-Party) vs. Jini (3-Party)



Effectiveness: UPnP (2-Party) vs. Jini (3-Party)



Efficiency: UPnP (2-Party) vs. Jini (3-Party)



Plan for the Next Six Months

- Submit two papers on recent results: MILCOM 2002 and 3rd International Workshop on Software Performance

- Complete characterization of UPnP and Jini behavior *(ending Phase I)*
  - Information propagation during message loss
  - State recovery during node failure
  - Performance under increasing network size

- Develop and document ideas for initial set of self-adaptive discovery mechanisms *(beginning Phase II)*

- Complete scalable (up to 500 nodes) discrete-event simulation model of UPnP – based on source code from Intel's Linux SDK for UPnP *(preparing our Phase II models for easy conversion to implementation during Phase III)*

- Extend our existing generic structural model of service discovery systems to cover behavior, message vocabulary, and consistency conditions *(we see this as a community service)*

Conclusions

- Emerging industry discovery protocols provide robustness properties through a division of responsibilities among: lower layer protocols, discovery protocols, and applications

- Characterizing the behavior and robustness of commercial service discovery protocols is a *necessary pre-condition* to developing and evaluating adaptation mechanisms intended to improve the performance of such protocols

- We described an approach to understand the behavior of service discovery protocols in the face of various faults: interface failure, message loss, and path and node failure

  (To learn more about the approach, see: "Analyzing Properties and Behavior of Service Discovery Protocols Using an Architecture-based Approach", C. Dabrowski and K. Mills, *Proceedings of Working Conference on Complex and Dynamic Systems Architectures,* sponsored by DARPA DASADA program)

- We applied the approach to characterize the performance of two different mechanisms (polling and eventing) for information propagation in various service discovery architectures (2-party and 3-party) and topologies (one and two SCMs) during interface failure

- We are currently conducting characterizations of performance in the face of message loss and node failures

Slides Containing Additional Details

Equating a Generic Structural Model of Service Discovery Architectures to Selected Commercial Discovery Systems

Generic Model	Jini	UPnP	SLP
Service User	Client	Control Point	User Agent
Service Manager	Service or Device Proxy	Root Device	Service Agent
Service Provider	Service	Device or Service	Service
Service Description	Service Item	Device/Service Description	Service Registration
Identity	Service ID	Universal Unique ID	Service URL
Type	Service Type	Device/Service Type	Service Type
Attributes	Attribute Set	Device/Service Schema	Service Attributes
User Interface	Service Applet	Presentation URL	Template URL
Program Interface	Service Proxy	Control/Event URL	Template URL
Service Cache Manger	Lookup Service	not applicable	Directory Service Agent (optional)

The Six Combinations of Architecture, Topology, and Consistency-Maintenance Mechanism Used in Experiments

Architectural Variant	Protocol Basis	Consistency-Maintenance Mechanism
Two-Party	UPnP	Polling
Two-Party	UPnP	Notification (with notification registration on SM)
Three-Party (Single SCM)	Jini	Polling (with service registration on SCM)
Three-Party (Single SCM)	Jini	Notification (with service registration and notification registration on SCM)
Three-Party (Dual SCM)	Jini	Polling (with service registration on SCM)
Three-Party (Dual SCM)	Jini	Notification (with service registration and notification registration on SCM)

Specific Division of Failure-Recovery Responsibilities Used in Experiments

Responsible Party	Recovery Mechanism	Two-Party Architecture (UPnP)	Three-Party Architecture (Jini)
Lower-Layer Protocols	UDP	No recovery	No recovery
	TCP	Issue REX in 30-75 s	Issue REX in 30-75 s
Discovery Protocols	Lazy Discovery	SM: announces every 1800 s	SCM: announces every 120 s
	Aggressive Discovery	SU: issues *Msearch* every 120 s (after purging SD)	SU and SM: issue seven probes (at 5 s intervals) only during startup
Application Software	Ignore REX	SU: *HTTP Get* Poll SM: Notification	SU: *FindService* Poll SCM: Notification
	Retry in 120 s after REX	SU: *HTTP Get* after discovery (retry up to three times) *Subscribe* requests	SM: depositing or refreshing SD copy on SCM SU: registering and refreshing notification requests with SCM
	Discard Knowledge	SU: purge SD after failure to receive SM announcement within 1800 s	SU and SM: purge SCM after 540 s of continuous REX

Significant Parameters and Values Used in Experiments

	Parameter	Value
Behavior in both two- and three-party architectures	Polling interval	180 s
	Registration TTL	1800 s
	Time to retry after REX (if applicable)	120 s
UPnP-specific behavior for two-party architecture	Announce interval	1800 s
	Msearch query interval	120 s
	SU purges SD	At TTL expiration
Jini-specific behavior for three-party architecture	Probe interval	5 s (7 times)
	Announce interval	120 s
	SM or SU purges SCM	After 540 s with only REX
Interface failure parameters	Failure incidence	Once per run for each node
	Failure scope	Transmitter, receiver, or both with equal likelihood
	Failure duration	5% increments of 5400 s from 0 to 75%
Transmission and processing delays	UDP transmission delay	10 us constant
	TCP transmission delay	10-100 us uniform
	Per-item processing delay	100 us for cache items 10 us for other items

Console Output from a Sample Experiment Run

```
Rate - 5
Run number - 21

SM 1  OUT Interface        down 365, up 635

SCM 1 OUT Interface        down 2417, up 2687
SCM 2 IN & OUT Interface   down 519, up 789

SU 1  IN Interface         down 2238, up 2508
SU 2  IN Interface         down 3256, up 3526
SU 3  IN Interface         down 207,  up 477
SU 4  OUT Interface        down 2876, up 3146
SU 5  IN Interface         down 4478, up 4748

Performance:

  SM  1 346.00000 346.00000 6 17
  SCM 1 346.00000 346.00016 61 102
  SCM 2 346.00000 346.00015 61 105
  SU  1 346.00000 346.00109 0 11
  SU  2 346.00000 346.00109 0 11
  SU  3 346.00000 5400.00000 4 11
  SU  4 346.00000 346.00109 0 11
  SU  5 346.00000 346.00114 0 11
```

Update Responsiveness (R)

Let D be a deadline by which we wish to propagate information to each service user (SU) node (n) in a service discovery topology.

Let t_C be the creation time of the information that we wish to propagate, where $t_C < D$.

Let $t_{U(n)}$ be the time that the information is propagated to SU n, where n = 1 to N, and N is the total number of SUs in a service discovery topology.

Define information-propagation latency (L) for an SU n as:

$$L_n = (t_{U(n)} - t_C)/(\max(D, t_{U(n)}) - t_C).$$

Define update responsiveness (R) for an SU n as:

$$R_n = 1 - L_n.$$

Update Effectiveness (U)

Let the definitions related to Update Responsiveness, R, hold.

Let X represent the number of runs during which a particular service discovery topology is observed under identical conditions.

Recalling that N is the total number of SUs in a service discovery topology, define the number of SUs observed under identical conditions as:

$$O = X \cdot N.$$

Define the probability of failure to propagate information to an SU as:

$$P(F) = (\text{count}(R_{i,j} == 0))/O, \text{ where } i = 1..N \text{ and } j = 1..X.$$

Define the Update Effectiveness for a given set of conditions as:

$$U = 1 - P(F).$$

Update Efficiency (*E*)

Let the preceding definitions associated with Update Responsiveness and Update Effectiveness hold.

Let *M* be the minimum number of messages needed to propagate information from all SMs to all SUs.

Let *S* be the observed number of messages sent while attempting (failures may occur) to propagate information from all SMs to all SUs in a given run of the topology.

Define average Update Efficiency as:

$$E_{avg} = (\mathrm{sum}(M/S_k))/X, \text{ where } k = 1..X.$$

Summary Statistics for Performance of Each Combination on Each Metric

	Mean (across all interface-failure rates)		
	Median Responsiveness	Effectiveness	Average Efficiency
Two-Party Notification	0.663	0.921	0.212
Two-Party Polling	0.615	0.973	0.251
Three-Party Notification (Single SCM)	0.601	0.894	0.389
Three-Party Polling (Single SCM)	0.530	0.911	0.201
Three-Party Notification (Dual SCM)	0.655	0.942	0.221
Three-Party Polling (Dual SCM)	0.587	0.927	0.110

95% C.I. for Each Metric-Combination at Selected Failure Rates

	Responsiveness			Effectiveness			Efficiency		
	5%	40%	75%	5%	40%.	75%	5%	40%	75%
Two-Party Notification	1.000 1.000	0.561 0.783	0.111 0.162	0.970 0.977	0.954 0.966	0.709 0.787	0.354 0.467	0.065 0.220	0.031 0.354
Two-Party Polling	0.975 0.980	0.501 0.849	0.076 0.138	1.000 1.000	0.993 0.993	0.760 0.826	0.501 0.666	0.031 0.230	0.042 0.059
Three-Party Notification (Single SCM)	1.000 1.000	0.605 1.000	0.042 0.095	0.993 0.993	0.939 0.955	0.521 0.652	0.827 1.000	0.099 0.504	0.033 0.320
Three-Party Polling (Single SCM)	0.974 0.980	0.244 0.412	0.043 0.083	1.000 1.000	0.946 0.960	0.660 0.753	0.387 0.512	0.043 0.173	0.040 0.164
Three-Party Notification (Dual SCM)	1.000 1.000	0.562 1.000	0.099 0.143	0.970 0.977	0.977 0.983	0.730 0.803	0.335 0.599	0.035 0.290	0.009 0.096
Three-Party Polling (Dual SCM)	0.974 0.986	0.391 0.543	0.056 0.096	1.000 1.000	0.939 0.955	0.660 0.753	0.218 0.273	0.033 0.103	0.019 0.059